ISO/TC184/SC4/WG3/N778

Third draft – 17/Feb/98

PRODUCT LIFE CYCLE SUPPORT

Statement of Technical Requirements (STR)

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Document Status

In the present form, the PLCS Statement of Technical Requirement is a working draft and represents a snapshot of ongoing work. Some parts may still be under development while for others the review process has already been started.

1.2 PLCS objective

The PLCS Initiative seeks to reduce life cycle costs, and improve product availability by improving the quality and accessibility of the technical information needed to support a complex product in operational use.

1.3 Document Objective

This document establishes a Statement of Technical Requirement (STR) for the PLCS Initiative for use by WG3/T8 and the Steering Committee in tasking the Program Manager to do the technical development work. Business issues (costs, benefits, organization, funding etc.) will be addressed elsewhere.

The STR extends the technical baseline established by the PWI Interim Report[1] and will, in time, replace it.

The document is expected to form an Annex or reference document for the IMS project proposal. The document will also be offered to ISO/TC184/SC4 for approval by WG3/T8 and WG3.

1.4 Document Structure

The STR presents information at several levels of detail.:

Section 2 defines a context for PLCS, in terms of the **types of product** to be supported and the characteristics of **the business environment** that are addressed.

Section 3: establishes the scope of the information to be addresses by the potential standards for the four business areas: The scope for Maintenance, Supply Chain, CM/Change Control and Support Engineering is defined in this section.

Section 4 defines the **information content** of the proposed standards and their intended **method of use**.

Section 5 further analyses the **business activities to be supported** based on the "AP architecture" diagram from earlier work.

Section 6 presents the more detailed **information requirements** to be addressed by PLCS in the form of English language statements and related graphical presentation. Readers not familiar with the background may wish to read Section 4 first.[2]

2 CONTEXT

This section describes the business context that the PLCS standards will be designed to support. It is written to help potential participants in the PLCS Initiative assess the relevance of the work. Extensions or changes to the context may require amendment of the STR.

2.1 Context for Product

The products of interest to PLCS are mechanical, electrical and electronic assemblies, developed to meet a customer need, that require some level of maintenance to sustain their operational use. PLCS is likely to deliver most value to the owner of major, engineered assets such as aircraft, process plants, communication systems, ships, vehicles and complex buildings that require significant development and have a lengthy in-service support phase.

The standard will also address the needs of smaller, simpler products that need to manage data relating to in-service use. A listing of industries likely to benefit from PLCS is attached at Annex C.

The products supported by PLCS may be:

- Fixed or mobile - a building or Main Battle Tank,
- Single or multi use - a telephone exchange or a multi- role aircraft,
- Single or multiple builds - an oil rig or a washing machine.

The products supported by PLCS will usually be defined by a configured product design. The design may have multiple variants. Each design variant may result in many physical product instances. The product design may be subject to change through life.

Products may be operated and supported with other, different products, as part of an operational fleet. The PLSC Initiative will focus on the requirement to manage data that derives from, or can be linked to, a single parent design (e.g. the C130 transport aircraft).

to different elements of the product – e.g. maintain on failure, maintain on elapsed time, maintain on condition.

The required maintenance may vary with the usage or operating scenario. Different types of maintenance response may be required e.g. repair in-situ, repair by replacement, strip-replace components-reassemble-test. Specialised tools and facilities may be required to undertake some maintenance tasks.

The work done, parts used and time taken in maintenance may need to be recorded to optimise support activities. Operating and maintenance history of a product instance may be needed to decide the appropriate maintenance action. The maintenance action may also depend on the "time to supply" necessary spares and tools.

Changes to the product instance configuration (modifications) may be implemented as a maintenance task. Operating and Servicing Tasks may also be treated as maintenance (e.g. check tyre pressure, fuel the aircraft, tow from hanger), as may activities to dispose of the product, or product components.

Different parts of the product may have different disposal strategies. Some parts may be subject to documented disposal procedures. Some parts returned for disposal may be re-cycled for re-use on the product, with or without refurbishment.

2.3 Context for Supply Chain Management

Supply chain management includes the acquisition and supply of components needed to build, modify or maintain the product during manufacture and in-service use. PLCS assumes that some components will be "bought in" from other suppliers. Selected components (spares) will be purchased and stored to support production and maintenance activities. These stocks may be managed separately. Components may be available from many different suppliers. The supplier of any component may change over time.

Different supply chain management strategies may apply to different components both during manufacture and during subsequent use e.g. purchase on-demand, Just-in-Time, lifetime buy, hold a minimum stock, hold machining instructions as AP224 file (BAMP)

2.4 Context for Configuration Management and Change Control

The PLCS Initiative assumes that formal procedures for Configuration Identification (CI) and Configuration Control (CC) will be applied to some, or all, of the product design, and, where appropriate to specific products instances.

The CI and CC system may also be applied to related, parts, tools procedures, and documents, including the product requirement statement.

Formal procedures for CI and CC may also be applied to some product instances (i.e. the control of serial numbered items).

The product design may support a range of permitted product instance configurations that can be altered in-service, after initial manufacture (e.g. configure for mission, optional modifications for customer use).

PLCS requires that a common set of Configuration Management concepts and definitions be applied throughout the lifecycle and across the supply chain. Different participants in the supply chain may apply different levels of control to different parts of the product.

2.5 Context for Support Engineering

PLCS assumes that a support engineering activity is conducted, in parallel with, or as part of the product design activity, to develop the strategy, policy and procedures for maintaining the product in service, and for establishing and sustaining the in-service logistic supply chain. The maintenance and supply concepts may be optimised over time by measuring support performance to compare achievement with expectations.

The support engineering activity will produce the information, tools and facilities needed to sustain the product use, based on supportability characteristics (life, time between overhauls, maintenance man-hours per flying hour etc) derived from the product requirement.

Product support activities may cross many organisational boundaries (e.g. operator, maintainer, suppliers, design authority, manufacturer, regulator etc.). PLCS requires

3 INFORMATION SCOPE

The information content of the PLCS standards, and any scope extension, will be set by reference to the prime objective: sustaining operational use. The AP Content diagram presented in the PWI Interim Report provided an initial scope definition.

The expected information content of each of the four business areas has now been further developed to provide an improved definition of the content of the proposed standards. Key elements from the Core are developed further in Section 5. An additional heading (Other) has been added to this section, to pick up items not addressed elsewhere.

Further development of the scope is to be expected as the project proceeds. Changes to scope will be approved by the Steering Committee.

The PLCS information scope is further defined as follows:

3.1 Maintenance Information

A first objective for PLCS is to define and structure the information needed to prevent or respond to predictable failures of the physical product instance, in a specified usage scenario. This information includes:

- the identification and configuration of the product instance including its physical and functional breakdown;
- the required product performance, to the level needed for maintenance;
- failure modes and diagnostic characteristics;
- relevant usage and operating history;
- maintenance task descriptions and associated resources (parts, tools, skills and facilities);
- a sufficient description of the product to support the maintenance activity (e.g. drawings, video clips etc);
- test and calibration procedures, following product maintenance;
- information on the return or safe disposal of items no longer required;

- the information needed to procure parts, including details of alternative suppliers
- the planned location of spares holdings.

The extent to which behaviour of the supply chain will be modelled (spares demands, delivery forecasts, transportation methods, actual stock levels, present location etc.) has yet to be decided[3].

3.3 Configuration Management

Changes to the product configuration may change the maintenance and spares required. Implementation of a change may be a maintenance task. PLCS will address the information needed to manage changes to the configuration of an existing product.

In addition, because of the high importance of the configuration structure to product information management, PLCS will also address the information needed to develop a configuration structure as part of the design activity and to manage changes to design.

PLCS will also provide a capability to determine what related items and information objects may be affected by a proposed or actual change.

Further details of the approach to Configuration Management are provided in Sections 4, 5 and 6.

3.4 Support Engineering

PLCS will define the information needed to develop and to optimise the support solution for the product, initially and during life. The information needed to achieve this includes:

- Intended operating and maintenance scenarios
- Supportability characteristics (required, forecast and actual)
- Classification of maintainer skills
- Policies and procedures for maintenance and supply
- Intended support solutions (the required maintenance and supply activities)

3.5.1 Feedback Required from Maintenance

PLCS will provide means to define the feedback that the maintainer is required to provide, including details of actions undertaken and reports on the state of the product. This will enable the capture of data on failure rates, repair times, spares usage, man hours etc. as an input to Life Cycle costing.

PLCS is not at this point intending to provide a full capability to model life cycle costs, or to address issues of price or monetary value, except in relation to the purchase price of a spare.

3.5.2 Maintainers Reports and Requests

PLCS will provide a capability to communicate reports and requests from maintainers regarding problems or suggested improvements to product design or support.

3.5.3 Approval Data

PLCS will provide a capability to record the approval status of products and related information in terms of who approved, for what purpose, when and (if necessary) why.

3.5.4 Information Objects

PLCS will provide a capability to hold any kind of information object, related to the product structure, to the level of capability provided by the NATO CALS Data Model. Version 3.0.

4 USE OF THE PLCS STANDARDS

4.1 Sharing and Exchange

The information model developed by the PLCS Initiative must be suitable for use in two different ways:

- For implementation, in part or in whole, as an integrated product support database, providing shared access to life cycle data (the information sharing or data warehouse approach).
- To provide a neutral format for exchanging agreed subsets of data between different databases, containing equivalent information (the information exchange approach).

A combination of these two approaches is likely to be needed in most practical cases.

4.2 Functionality of a Shared Database

The PLCS initiative will not attempt to define the functionality to be provided in an integrated product database. In a shared access environment, decisions on how much of the PLCS data model to implement, whether to add additional information and what functionality to provide, will be left to the discretion of implementers.

4.3 Exchange Capability

The PLCS Initiative will develop a set of Interchange Specifications, to define some standardized subsets of the data model for use as neutral exchange formats. The relationship of these Interchange Specifications to the existing and emerging STEP formats – APs, Conformance Classes, Modules, Part 21 Files, SDAIs etc. – has yet to be determined.

A methodology will also be provided for establishing additional user defined Interchange Specifications. Consideration will be given to developing a registration mechanism.

data to enable the status and significance of the exchanged data to be understood. This will address all stages of the product its life cycle: as REQUIRED, as DESIGNED, as BUILT and as MAINTAINED (See Section 6.1)

PLCS will also provide a capability to attach any kind of information object to the product structure, with appropriate linkage to the physical or functional breakdown, and to other related data elements.

4.3.2 Supportability Characteristics

PLCS will provide a capability to communicate a defined set of supportability characteristics of the product (reliability, mean time to repair etc.), including the de-composition and allocation of targets to components of the product, based on a physical or functional breakdown.

The same structure will also be able to communicate other types of requirement (e.g. performance capability) although these may not be modelled in detail. Opportunities for co-operation with the System Engineering AP project New Work Item (SC4 document N674) will be actively explored for modelling system requirements.

4.3.3 Usage Scenarios

PLCS will provide a generic capability (not detailed modelling) to communicate different operating and usage scenarios related to a product.

4.3.4 Skill Levels

PLCS will provide a capability to establish and communicate a classification of maintainer skill levels, supported by textual description.

4.3.5 Predicted Failures

PLCS will provide a capability to describe and exchange information related to predicted failures of the product and its components. This will be achieved by developing entity-attributes to relate product failure modes, failure consequences and failure diagnostic information to elements of the product breakdown (functional or

4.3.7 Maintenance Feedback

PLCS will provide a capability to define the nature and extent of feedback information which maintainers are required to provide.

4.3.8 Maintainer Reports and Requests

PLCS will provide a capability to capture and communicate reports of requests from maintainers, regarding problems, issues or improvements to the product support and design

4.3.9 Parts Properties and Classification

PLCS will provide a capability to capture and exchange the information needed to identify and describe all parts of the product to be held as spares. Opportunities for using ISO 13584-42 (PLIB) will be fully explored.

4.3.10 Intended Spares Holdings

PLCS will provide a capability to define the intended holdings and location of spares required for maintenance.

4.3.11 Product Operating and Maintenance History

PLCS will provide a capability to hold and exchange specified aspects of product operating history of direct relevance to maintenance (e.g. hours run). Opportunities for extending capability in this area will be explored with the Japan/Korea PWI Process Plant Operations & Maintenance project (SC4 document N278).

5 BUSINESS AREA ANALYSIS

This section provides further analysis of the business and information requirements from the viewpoint of the four business areas identified by the AP content diagram.

5.1 Support Engineering

5.1.1 What is Support Engineering?

The objective of Support Engineering is to maximise the operational availability of a product while minimising the through life cost of ownership of a product. Since much of that cost is “locked in” during the design phase, it may be worth investing significant effort, from the earliest stages of development, to minimise through life costs. This can be achieved by improving the product design and by effective design and planning of the intended means of support.

Once the product has entered service, planned provision of support can be based upon the predicted reliability and expected life of its component parts. In-service experience will validate or challenge these predictions. Where predictions are found to be significantly in error, it may be cost effective to initiate changes, either to the product design, or to the support system, to control or eliminate the emergent causes of poor availability or high support costs.

The requirement for Support Engineering can, therefore, best be defined by considering these phases of the product life cycle separately.

5.1.2 Support Engineering during Product Development (Initially and for changes).

5.1.2.1 Requirement Capture

The Support Engineer needs a clear statement of the functional requirement for the proposed product from which a performance metric can be derived in order to verify operational acceptability. This statement must be sufficiently precise to enable

5.1.2.2 Operating Context or Usage Scenario

To optimise the support solution, the Support Engineer will require a set of assumptions for the expected usage of the product. These may be recorded, for use in optimisation algorithms. The Operating Context will define such details as:

- nature of the operational environment;
- length of the operating cycle between support opportunities;
- capability and availability of support staff;
- typical pipeline times for the delivery of spares from the various storage points in the supply chain;
- capabilities of the existing support infrastructure.

Definition of an expected usage scenario does not preclude use of the product in a different context. It will however provide the owner with a clear understanding of the basis on which the support solution was develop and provides a basis for reworking the support solution, to reduce costs and improve availability, if the operating context changes.

5.1.2.3 Failure Modes Effects and Criticality Analysis

During the design phase, the Reliability Engineers will seek to eliminate or minimise the potential for the product to fail. There will, however, always be a residual number of failures that it is neither technically possible nor cost effective to eliminate. Each one of these remaining predictable failures should be supported by a maintenance strategy to reduce the risk of failure to acceptable levels and to provide a remedy should failures occur. As the failure may occur well into the life of the product, there is benefit in retaining a record of the failure analysis, and the reasoning behind the related support solution.

5.1.2.4 Task Analysis

The maintenance strategy adopted is likely to comprise a series of condition monitoring, scheduled, occasional and exceptional tasks. Each of these must be analysed to

5.1.2.6 Task Association

The tasks identified during the task analysis may be generic to the failure mode. To create a viable maintenance solution the maintenance task must be associated with the components of the product to which they apply. For example the failure mode “fails to inject fuel due to injector blockage” may lead to a task “clean the injector”. This task can then be associated with each of the injectors fitted to the product, as uniquely identified within the product configuration record.

5.1.2.7 Spares Analysis[6]

The acquisition and storage of spare parts is a major cost driver. The Support Engineer will therefore strive to keep stockholdings of spares to a minimum by matching recommended holdings to the predicted take off derived from the maintenance strategy adopted. Various algorithms exist to assist with this process. The conclusions are very sensitive to changes in input parameters such as component Mean Time Between Failure estimation, logistic pipeline delay time or component remanufacture lead time. The analysis results will have to be revisited whenever algorithm parameter values are updated or the given usage pattern of the product is changed.

5.1.3 Support Engineering In-Service

5.1.3.1 Failure and Discrepancy Reporting

Support Engineering during development will require much engineering judgement, including the interpretation of data from similar products for which operational experience is available. To optimise support performance, and minimise in service costs, data should be collected to check whether the support system is working as intended, or as currently required. Defining the required feedback is part of Support Engineering. Feedback can be grouped under two headings:

5.1.3.2 Failure Reporting and Corrective Action System FRACAS

- Normal Reporting. Where failure occurrence and recovery action conform to predictions, reporting may be limited to a report that a failure has occurred.
- Intensive Reporting. Where more detail is required, either because insufficient data was available during development to predict in-service behaviour with confidence, or because the failure rate, spares usage or maintenance costs are higher than expected, a period of special reporting may be justified. This may include the reporting of additional information related to failure occurrences (e.g. the hours run prior to failure), more details of the failure itself or the tracking of replaced components through a repair loop to correlate information from strip reports with failure instances. Intensive reporting may also be required early in the equipment life to confirm performance against contract requirements. The nature of the feedback required will be specified on a case by case basis.

5.1.3.3 Discrepancy Reporting and Corrective Action System DRACAS.

Although the Support Engineer will have made every effort to predict and provide solutions to all the potential failure modes of the product, provision must be made for when this process itself has failed. Three levels of discrepancy are recognised:

- Analysis Errors. In-service experience will identify where the planned procedure for managing an anticipated failure mode has failed to provide an accurate or complete solution. Response to reported errors may vary from a minor adjustment, such as changing the planned duration of a task, to a major reworking of the response to the particular failure.
- Additional Failures. A discrepancy may report additional failure modes, not identified during development, but revealed by in-service experience. These will need to be analysed and maintenance strategies generated or amended accordingly.
- Stores Usage[7]. Inventory management procedures should detect when spares usage does not match expectations. Such a discrepancy will bring into question the validity of the original Support Engineering analysis and may require a

- what has been designed
- what has been built
- what has been, or should be, changed
- what related items are applicable to the product

Accurate, timely information concerning the configuration of a product and its related items is needed throughout the life cycle to achieve cost-effective manufacture, continued safe operation and efficient in-service support. However, the level of effort applied to Configuration Management will vary with business circumstance. In some applications it may prove necessary to control the configuration of each instance of the product to a high level of accuracy to ensure safe operations (e.g. flight safety, nuclear plant). In other circumstance, where errors can be more readily accepted, control configuration may have a more limited scope.

PLCS must provide capability to define and manage the configuration of complex items, over their full life cycle, to the serial number level. The level to which configuration management is applied to any specific product will remain a business decision.

The functions of CM include Configuration Identification (CI), Configuration Change Control (CC), Configuration Status Accounting (CSA) and Configuration Audit (CA).

5.2.2 **Concept and Definitions**

5.2.2.1 Configuration Identification (CI)

The CI process establishes a comprehensive system of identification for a product and related items. CI answers the question "What is the product" to the level required by the business context in question. All products whose configuration is to be managed are assigned unique identifiers. This is done to enable one product to be distinguished from another; one assembly of a product to be distinguished from another assembly and to match related items to the product or assembly to which they apply. Identification may

5.2.2.3 Configuration Status Accounting (CSA)

CSA is the activity of maintaining an accurate record of product configuration. CSA records the current status of any product, its structure, the history of any change within an integrated and coherent information base.

5.2.2.4 Configuration Audit (CA)

CA verifies that the item is in compliance with its corresponding configuration information. CA answers the questions: "Is the item built according to the design? Does the item do what it is supposed to do?"

5.2.2.5 Configuration Item

Configuration Items are products to which an organization intends to apply Configuration Management control. They may also be defined as items to which control by version number will be applied.

The identification of the Configuration Items is dependent on viewpoints. Different life-cycle organizations may have different views over what configuration items they need to manage.

For example:

- the user may decide that he will control configuration of physical product instance while the product design configuration control will be the responsibility of the equipment supplier;
- the main contractor may decide not to maintain configuration control of items supplied by a subcontractor.

In any case, a standardized interface is needed to exchange consistently configuration information across different users of the system.

There are three different kind of products that may need to be defined as Configuration Items: (1) Physical Products, (2) Documents, and (3) Software. Configuration Items

structure; (2) a document (e.g. Technical Manuals) may contain Configuration Items, either as sub-documents, or as physical product (e.g. a CD Rom). The conceptual state of a document is a digital file or a set of digital files. Its realised state is the representation on paper or on a computer screen using the appropriate application (CAD programs, SGML or HTML Browser, Video Player, DBMS Query). A document may have more than one document representation and some of these may be electronic while others are not. All document representations of a particular conceptual document must have the same information content, although the presentation format (both electronic and visual style) may vary.

- Software: '*A combination of associated computer instructions and computer data definitions required to enable the computer hardware to perform computational or control functions'*. Like documents, software may appear within a configuration structure and may itself be subject to the CM process. The conceptual state of software is the source code and the linked external library and component. Its realised state is the executable application and the related files.

5.2.2.6 Basic structural concepts

To fully define a complex product (Physical Product, Document or Software) it is often necessary to define the components from which it is made. In a generic structure we may identify one top element (the *root*), elements that are *nodes* of the tree and that are composed of other nodes, and basic elements, the *leaves*, that are not further decomposed. The tree structure is a directed acyclic graph (DAG), that is, a collection of nodes and directed links such that no node is an ancestor or descendant of itself.

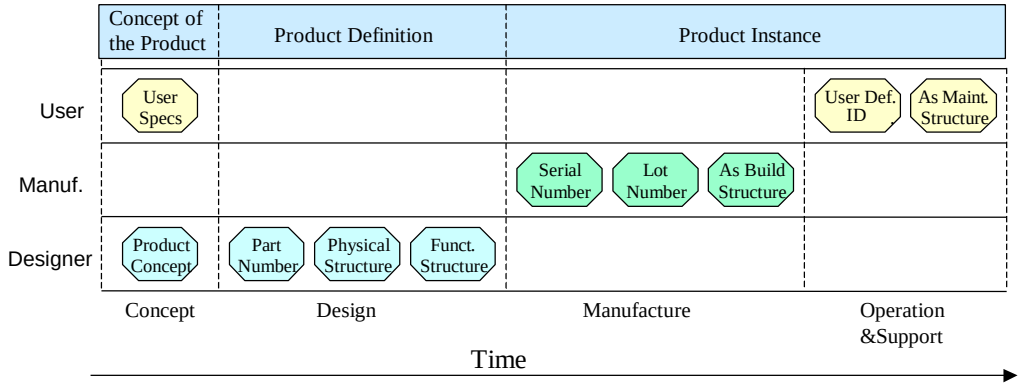
The content of a product structure is dependent on purpose, viewpoint and context.

For example, if a product structure is aimed to support the Bill Of Material Structure, the basic elements (leaf) may be the equivalent of Part as defined by IEEE Std 1220-1994: '*the lowest element of a physical or system architecture, specification tree, or system breakdown structure that can not be partitioned without destruction or*

5.2.3 Configuration Identification

5.2.3.1 Evolution of Product Identity

Product identification should reflect the product evolution through the complete life cycle. The figure below illustrates the main product identifiers that are normally issued during the life cycle of the same product.



Product identifiers during the life cycle are assigned by different organizations.

For example, the user organization normally assigns its own identifier (e.g. tail number, plate number) to End Items for the purpose of tracking them. User-defined identifiers (e.g. NATO Stock Numbers) are assigned to Parts and Assemblies when these items need to be managed by the user stock management system.

It is essential that links between different identifiers of the same product are consistently maintained throughout its the life cycle.

5.2.3.2 Three Layer Model for Product Identification

To deliver effective CM it must be possible to uniquely identify all products.

The third and bottom level is the product id name (e.g. part number, serial number, file name). It is provided by the Organization and is assumed to be unique within the context of that Organization.

5.2.3.3 Identification of Physical Products

5.2.3.3.1 Conceptual State Identifiers

Part Number: It is an identifier consisting of an alphanumeric string assigned by the design agency for an item and used to correlate the product design with the physical instances of the product.

5.2.3.3.2 Realised State Identifiers

Part and assembly unit / batch identification. Each physical part or physical assembly must be identified by marking it with the part number (design identification). If it is critical to safety, performance, or operation, each individual part or assembly must also be identified by a unique product tracking identifier (serial number and/or lot number). In some cases, more than one type of product tracking identifier may be assigned (e.g. serial number and lot number); in this case, the correlation between product tracking identifiers should be maintained.

- Serial Number: It is an identifier consisting of alphanumeric characters which is assigned sequentially in the order of manufacture or final test and which uniquely identifies a single item within a group of products manufactured from the same design.
- Lot number: It is an identifier consisting of either a date or a string of alphanumeric characters which uniquely identify a group of units of the same product which are manufactured or assembled by one producer under uniform conditions and which are expected to function in a uniform manner.

User-Defined Identifier: The user may need to issue additional identifier to End Item (e.g. tail number, plate number) in order to manage them within the user management system. In addition, it may be necessary to issue additional user-defined identifiers to

5.3.2 Background

The objective of a Maintenance Management system (MMS) is to deliver the required product availability by implementing the support solution derived through the Support Engineering activity.

Maintenance tasks can be classified according to type. The terms ‘servicing’, ‘preventive’ or ‘corrective maintenance’ provide one such classification. Current maintenance concepts have added new terms such as ‘condition monitoring’, ‘scheduled’, ‘occasional’ and ‘exceptional’. This document uses the latter terminology with the assumption that ‘preventive’ can be mapped to ‘scheduled’ and ‘corrective’ can be mapped to ‘exceptional’.

5.3.3 Implementing a Maintenance Management System

5.3.3.1 Data Take-on

The first task, when commissioning a maintenance management system, is to establish the information baseline for the selected support solution that is relevant to the product instances on which maintenance will be performed. As the support solution may evolve over time, periodic updates of the maintenance management system may be needed to implement revisions. This PLCS development must support a managed evolution from current systems to ensure that no information is lost about maintenance or usage history, or product condition.

5.3.3.2 Configuration Reconciliation

A reconciliation may be required to eliminate inconsistencies between the ‘as designed’ configuration used by the support engineer to develop the support solution and current configuration found by the maintenance manager. In many cases it will also be necessary to maintain a configuration history

5.3.4 Maintenance Execution

It may also be necessary to record the service and maintenance history of components that are removed from a higher assembly, refurbished, stored and re-installed into a different location (i.e. a 'log card' for the component).

5.3.4.2 Types of Task

5.3.4.2.1 Servicing[8]

Servicing routines are undertaken as part of the operating cycle of the product. They cover such tasks as fuelling and lubricating and, while clearly part of the maintenance schedule, are not normally planned as part of the maintenance programme. They do not generally associate with the FMECA. Where an association has been made, the routine would probably be better classified as a 'condition monitoring' task.

5.3.4.2.2 Condition Monitoring

Certain components within the product will reveal measurable wear parameters. Condition Monitoring tasks, identified during Support Engineering, may require these parameters to be measured and recorded at the appropriate interval to detect deterioration. Such measurements will need to be scheduled by the maintenance management system. Data may be collected automatically if an appropriate sensor is installed. Results of condition monitoring must be compared with pre-determined values to detect when wear thresholds have been breached. Out-of-tolerance values should cause the MMS to add the appropriate remedial occasional tasks into the work scheduler for action before failure occurs.

5.3.4.2.3 Scheduled Tasks

Certain components will be assessed as having a finite life, attracting a routine tasks to replace or refurbish them before end-of-life. Where life is measured as a linear unit of time, the work scheduler can programme these in the schedule from the date last done. Where the measure is more complex, such as 'number of starts', an alternative means of scheduling must be invoked.

5.3.4.2.4 Occasional Tasks

Occasional Tasks are the pre-planned actions necessary to recover from the impending

5.3.4.2.5 Exceptional Tasks

When failures occur that have not been anticipated through the support engineering activity, or work of a non routine maintenance origin (such as a change of paint scheme) is required, then the MMS must be capable of capturing the work requirement and managing it through the common work scheduler. It is particularly important to capture as much information as possible when an unexpected failure has occurred, since this represents a failure of the support engineering process to anticipate the failure and prepare a solution. In addition to providing management control of the remedial activity, the MMS should also generate a discrepancy report, as outlined below.

5.3.4.3 Task Management

5.3.4.3.1 Task Description

Within a task description, there must be sufficient detail on what is to be done to allow a technical cost centre to quote a fixed price for the work. The Task Description can take any format or combination of formats, from simple text through to a video clip with a sound track. The list of parts required to execute the task must be separately identified and accessible to external processes, such as automatic stores ordering when the task is scheduled.

5.3.4.3.2 Task Scheduler

The task scheduler within the MMS is a management tool for programming work known to be due and then allocating it to appropriate staff for execution. As a described task may be common to one or more components within the product, the scheduler must be able to associate the due task with the particular instance of the component against which the task is required. In addition to scheduling work on a day by day basis, as required by the different task natures already described, it must also allow the manager to package one or more of the outstanding tasks into requests for external assistance. When a task has been scheduled, the MMS should be capable of demanding all the necessary resources, such as spares, external facilities and specialist skill teams as necessary

5.3.4.4 Inventory Management Interface

undertake) then the system must be able to communicate a request for assistance to an external support authority. If that authority has access to the support solution data, this could take the form of a simple pointer to the relevant task records but where this access is not available, then a more complete transfer will be required.

5.3.5 Feedback[9]

5.3.5.1 Failure Reporting and Corrective Action System (FRACAS)

At its simplest, the MMS will collect data on the material state of the product. This will take the form of parameter values from condition monitoring, records of maintenance tasks completed and occurrence information on actual failures. Provided the recovery from failure runs according to the prediction contained within the 'occasional' task description, this information serves to validate the support engineering analysis and to provide statistical data to the product owner.

5.3.5.2 Discrepancy Reporting and Corrective Action System (DRACAS)

Before any instance of a task is confirmed as complete, the maintainer should confirm that the task description gave a full and correct account of the task. Where any part of the description was found to be in error or inadequate, then the maintainer should record the discrepancy to trigger a review by the support engineering activity. Revisions will be promulgated as changes to the approved support solution.

5.4 Inventory Management

5.4.1 Part Definition

As a first priority within Inventory Management, PLCS should seek to define all of the product design data needed by an Inventory Manager (i.e. functional, physical and support design information) for all parts used by/in/for the system. This would include:

- Parts details, including alternatives, and known suppliers etc (sufficient for classification and unique identification within a Parts Library)

This would be achieved by the development of a set of interchange specs/conformance classes that can exchange the data needed.

5.4.2 Management Activity

PLCS should cover the Inventory Management activity for a single product (i.e. the information needed to manage, issue and account for an integrated inventory of parts specific to the single product). PLCS will not addresses the complexity of managing an inventory system supporting a mixed range of products. The information set includes :

- historic demand
- predicted future demand
- available funding
- minimum stock levels
- re-order quantities
- stock management policy (including storage locations)
- Electronic Commerce protocols i.e. a Logistics Supply Support Plan.

5.4.3 Management Assumptions

Supply Chain Management must address the issues of "pipeline times" in the supply chain. These include :

- The time taken to deliver a part, on demand, from the store location to the point of need
- The time taken to return a defective item for repair and receive it back into the pool of serviceable stock
- The time taken to remanufacture a part from source information (i.e. RAMP)

The Supply Support Plan should satisfy the spares support requirement determined during the support engineering activity. The solution will contain sparing decisions based upon a set of generic operational scenario assumptions that include pipeline time

6 INFORMATION REQUIREMENTS

This section presents the detailed information requirements developed for the PLCS initiative. It is a working document and represents a snapshot of on going work. The Express diagrams presented in this section are provided as visual aids to better describe the requirements and NOT as modeling solutions.

6.1 What is "PRODUCT"?

Defining a 'product' is complex because there are many ways of looking at the same thing. STEP defines a product as '*a thing or substance produced by natural or artificial processes*'. Use of the word '*produced*' in the above definition may communicate the idea that a product is always a *physically existing object*. This is not true. At the very early stage of its life cycle, a product may exist as simply as concept described by the customer needs and operational requirements. At the end of the design phase a product may be seen as *physically realizable object* or 'design'. It becomes a *physically existing object* only at the end of the manufacturing process. Once realized the product is to be *supported* throughout the life cycle.

To manage products over their life cycle it is necessary to address these different views of *product* and to distinguish between the product 'As Required', 'As Designed', 'As Built' and 'As Maintained'.

This can be accomplished as follows:

- the life-cycle of a product starts with the definition of a *product_concept* which is the idea of a product as defined by customer needs. The *'AS REQUIRED'* view of a product is defined by the mission need that created the demand for the product and is described by its required functionality (*product_requirement*: e.g. expected features, capabilities, performance, et cetera);
- the design activity, based on the required functionality, progressively defines the physically realisable objects or the 'AS DESIGNED' view of the product. The set of related designs is then used to manufacture a quantity, possibly thousands, of physical products (*product_instance*);

These objects are closely inter-related as shown in figure 1. Relationships should be maintained throughout the life cycle.

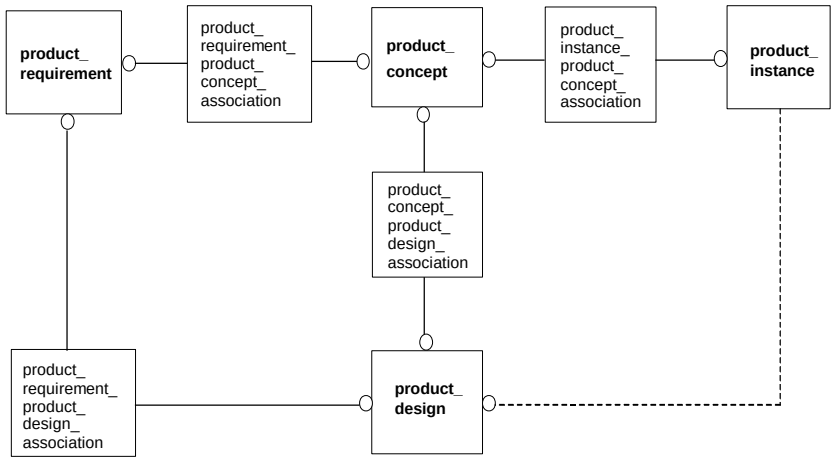


Figure 1 – The product architecture

The ***product_concept*** object is the collector or common root for all subsequent objects. The product concept is the idea of a product as defined by customer needs. It identifies a deliverable product as perceived by the customer (e.g. an item in the catalog of a supplier).

A *product_concept* may exist without a *product_design* or a *product_instance*. It is related to the object ***product_requirement*** which describes the required performance or behaviors of the deliverable product. The object *product_requirement*, in turn, is related to *product_design* making it possible to relate functionality to design.

The ***product_design*** object is the container of (1) functional design, (2) physical design and (3) support design. It is related to *product_concept* and to *product_instance*. The latter is the relationship between a specific actual object (identified, for example, by its serial number) to the design information from which it was developed. This relationship

6.2 The PLCS Perspective

The PLCS objective is " … *to improve product availability by improving the quality and availability of the technical information needed to support a complex product in-service*". The term *product* in this definition should be seen and understood as the *product_instance* described above (only a physically existing object may be supported).

For PLCS perspective the main focus will be placed on the *product_instance* object.

To support a *product_instance* in-service, we need to know its ACTUAL configuration, role, condition state, maintenance history and operational history (AS BUILT, AS MAINTAINED). Then we need a mechanism to link the product_instance to (1) the maintenance directive resulting from the support engineering activity; and (2) to the technical data resulting from the functional and physical design activity. In this proposed architecture the linking mechanism between *product_instance* and the other product objects is achieved through relationships.

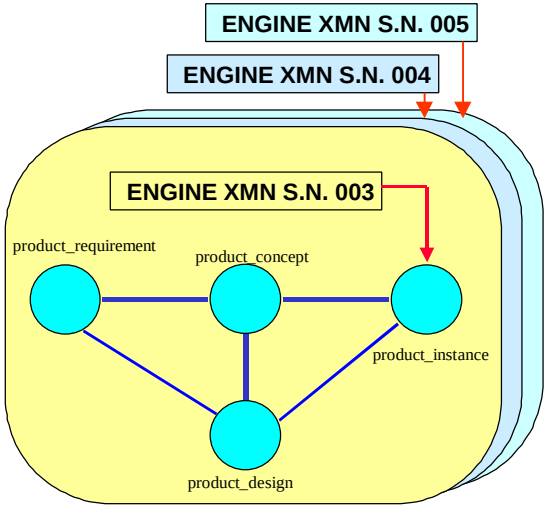


Figure 2 – product_instance relationship

6.3 Product Concept

A *product_concept* is the idea of a product as conceived by the user. It will often correspond to the items supplied to the user (e.g. an item in the catalog of a supplier). The definition of *product_concept*(s) is driven by user's needs and by user defined usage scenario. It represents the idea of a product based on user viewpoint and NOT as it might be designed or manufactured.

The basic relationships of *product_concept* are described in figure 3.

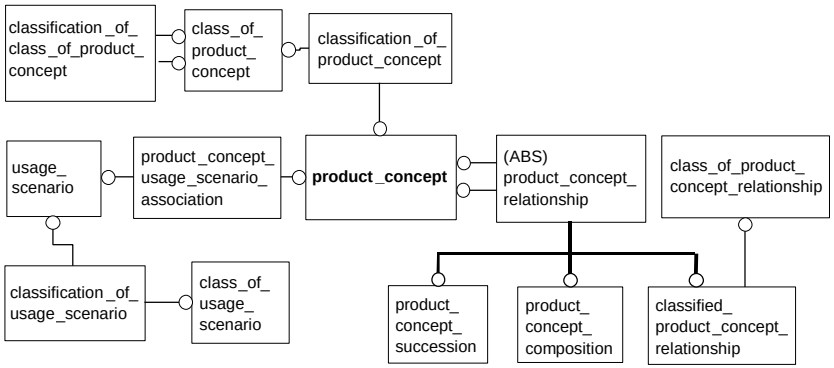


Figure 3 – product_concept basic relationships

Some detailed information requirements are the followings:

1. a *product_concept* is defined by a name, an identifier and a textual description;
2. the *product_concept* identifier is the combination the user organization identifier and a unique identifier assigned by that organization. It is assumed that the user organization issues unique identifier within its domain;
3. *product_concept*(s) may be classified, decomposed and aggregated. Class of *product_concept* should be defined[11];
4. *product_concept*(s) may be versioned and may be subject to Configuration Change Control (configuration item);

- user conditions (e.g. human capability and limitations, national laws and regulations);
- support conditions (e.g. support resources, pipeline times);
- mission phase (e.g. landing)

7. simple conditions may be combined with AND, OR and XOR operators to define complex usage_scenario(s);

8. *usage_scenario*(s) may be organized in classes. Most probable scenario and worst case scenario in peace-time and wartime employment are examples of *usage_scenario* classes.

6.4 Product Requirement

A *product_concept* in a specific *usage_scenario* may be characterized by a set of required or expected product features, performance or behaviors identified by the user or derived by the user's needs.

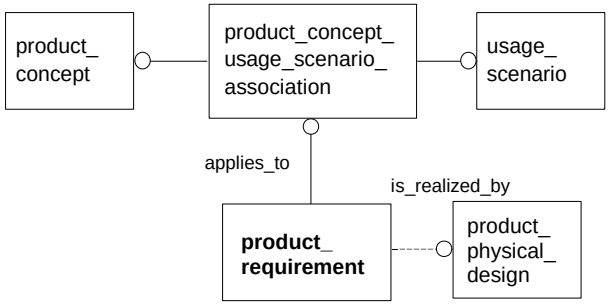


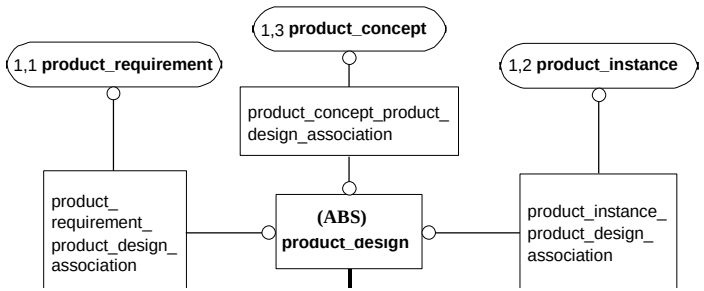
Figure 4 – product_requirement basic relationships

A *product_requirement* relates to one or more *product_concept_usage_scenario association* and to zero, one or more *product_design*. This implies that a *product_requirement* instance may exist without a *product_design* but it could not exist without a *product_concept* and the associated *usage_scenario*.[13]

3. classes of requirement may be defined. Some product requirement classes may be:
   - constraint definitions (*e.g. budget constrains, human limitations);*
   - functional requirements (e.g. *to provide an output voltage of 24 Volts plus-minus 0.5V);*
   - operation requirements *(to work ceaselessly for 2500 hours in usage_scenario n.2).* Includes also mobility, mission frequency and duration;
   - support requirements[14] *(to be repaired on site within 48 hours);*
   - physical requirements *(not more than 5 Kilos);*
4. change of a *product_requirement* should be addressed by configuration change control entities such as change request, implementation directive and applied solution (see paragraph 6.7.1.1);
5. associations between a predecessor *product_requirement* and a successor *product_requirement* should be maintained. This association states that the successor requirement is created by modifying the predecessor requirement;
6. *product_requirement*(s) may be described by using STEP IRs constructs such as property definition, property representation and measure schema;

6.5 Product Design

The *product_design* object has a number of relationships with the other views of the 'product' (see para. 6.1 and figure 5).



The product_design may consist of functional design, physical design and support design. These three components are related each other, making it possible, for example, to derive the *product_physical_design* instances that are involved in a function or the *product_support_design* instances applicable for a particular physical design.

6.5.1 Product Functional Design

The product functional domain describes what activities are performed, within a system, in one or more usage scenario, to fulfill a requirement.

The diagram in figure 6 describes the basic relationships:

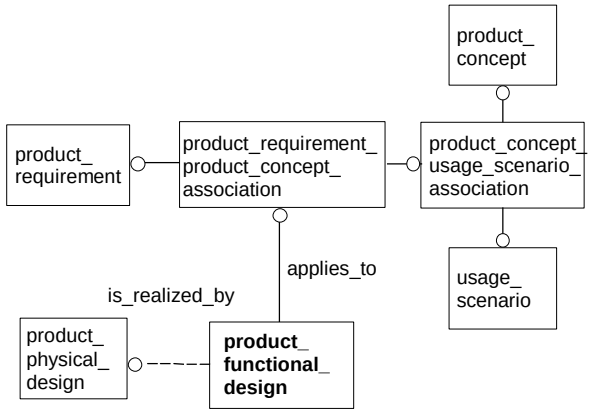


Figure 6 – product_functional_design basic relationships

A function could provide a service, process materials or change the state of the system environment. Functions form a hierarchy, with the top level being the function of the *product_concept* itself and the bottom level being individual actions.

For example if ‘Electric Generator’ is the *product_concept*, the functional hierarchy top level would be ‘Produce Electric Power’ with such sub-functions like ‘Provide Fuel to the Combustion Chamber’ which in turn could be decomposed into ‘Store Fuel’, ‘Supply Fuel’ and ‘Inject Fuel’. The functional analysis is useful to identify the physical

Functional design is not achieved solely by hierarchical decomposition of top-level function. In addition to hierarchical relationship functions may interact in many other ways. For instance, a function could be activated based on the result of a previous function. In this case a chain of functions is defined. In the above example the function 'Obtain Current Position' cannot be activated until the function 'Identify Target' has been completed.

A typical relationship in the chain is the one between a *caused_function* and a *causing_function* meaning that the first is affected by a status change of the second. In a wider sense, each function may be controlled by a behavioral interface or 'control-port' which affects its state.

Some of the data requirements for product functional design are the followings:

1. a *product_functional_design* relates to one or many *product_concept*(s) and to zero, one or more *product_requirement*. This implies that a *product_functional_design* instance may exist without a product requirement, but will always be related to at least one *product_concept*;
2. a *product_functional_design* is realized by zero, one, or more *product_physical_design*;
3. a *product_functional_design* is uniquely identified by the combination of the designing organization identification and by the identifier assigned by that organization;
4. a *product_functional_design* may be controlled by zero or one organization at a given point in time in one or more control methods;
5. control methods should be defined. Authority for approval is an example of control methods.
6. a *product_functional_design* may be categorized. A class of *product_functional_design* may in turn be organized in a class hierarchy;
7. agreed classes of *product_functional_design* should be defined;
8. a *product_functional_design* may relate to another *product_functional_design*;

13. whether a *control_port* is enabled is decided by the value and type of its attributes;

14. a *product_functional_design* has multiple representations in various formats such as plain text, structured text (SGML, HTML, XML), drawings, video, audio or external documents.

The diagram in the figure 7 covers most of the above requirements:

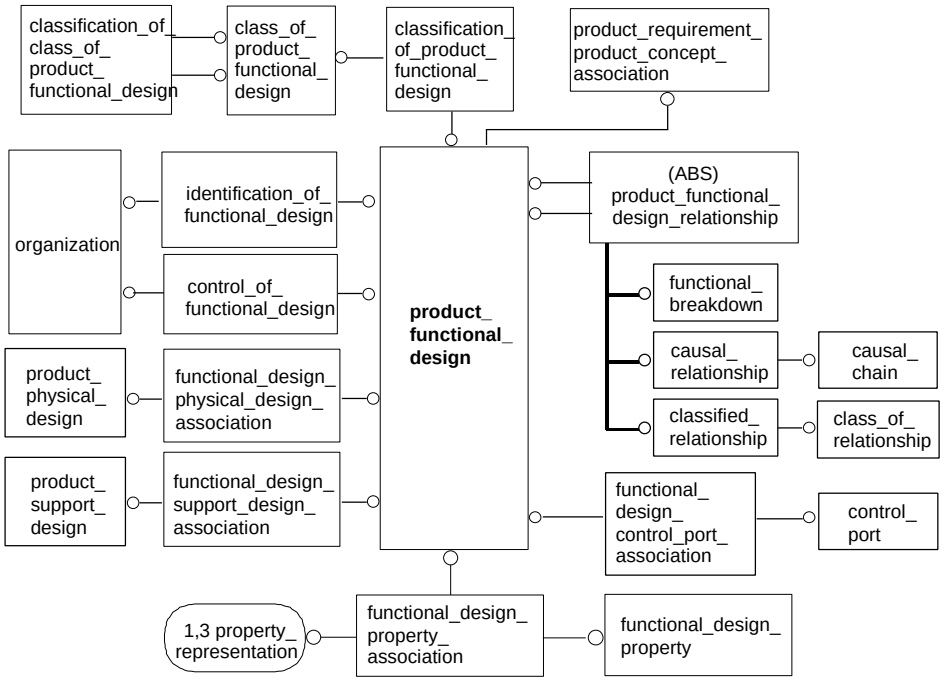


Figure 7 – product_functional_design high level model

6.5.2 Product Physical Design

Product physical design is an abstraction of *product instance* or the design of a

1. each *product_physical_design* is identified in the context of an organization. This implies that the same *product_physical_design* may have multiple identifications for different organizations.
2. different types of product reports should be possible by using the *product_physical_design*. These reports should be able to describe the product structure to many levels of details[16]. Level of details may address, for instance, the degree of decomposition (e.g. single level or multi level) and the type of decomposition (e.g. exploded, flattened, indented … etc);
3. Bill of Material, Part List, Illustrated Part Catalogue Structures are example of product reports;
4. a *product_physical_design* is characterized by zero, one or more properties;
5. *product_physical_design* properties may be represented using STEP IR constructs[17];
6. in addition, *product_physical_design* properties may be represented by plain text, structured text (SGML, HTML, XML), drawings, video, audio or references to external documents;
7. *product_physical_design* may be classified. Class of *product_physical_design* may in turn be organized in a class hierarchy;
8. a class of *product_physical_design* may have a set of predefined properties that must or may be filled-in to represent a *product_physical_design* instance that is a component of that class;
9. *product_physical_design* changes should be addressed by change process information such as change request, implementation directive and applied solution.

6.5.3 Product Support Design

In simple terms, the objective of support engineering is to determine what can go wrong

6.5.3.1 Failure Analysis

It is assumed that, at the end of the design phase, a residual number of predictable failures still exist. This occurs because it is either impossible or not cost effective to eliminate the failures or because they result from external agents. Each one of these predictable failures should be addressed by a maintenance strategy to:

- reduce the risk of failures to a minimum (preventive maintenance);
- provide a remedy should the failure occur (corrective maintenance).

A failure involves a *product_physical_design* while performing a certain function in a particular usage scenario.

More precisely, the predicted failure of a product applies to one or many *product_physica_design* and happens in the context of one or many *product_functional_design* and of one or many *usage_scenario*. (see figure 8)

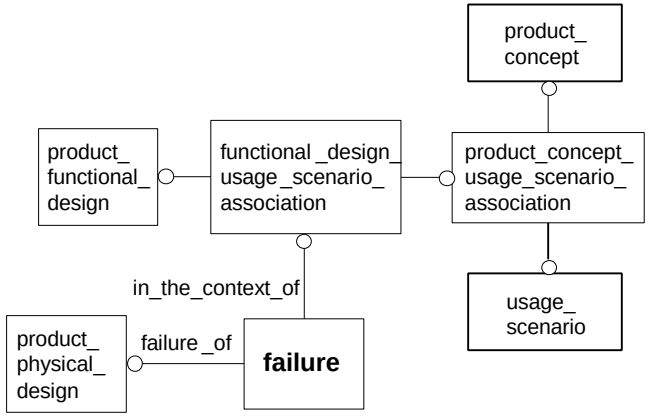


Figure 8 – failure[18] context

A failure is described by its:

- *Cause:* failure causes may be classified as internally generated within the system

In a more complex situation, figure 9, a particular failure (Failure 4) will occur only if two other failures (Failure 1 and Failure 2) occurs together or if a third (Failure 3) occurs by its own and not with the first two.

In a cause-effect chain it should be possible to combine failure causes with:

- AND operators, when the related causes must occur together;
- OR operators, when the related causes can but need not occur together;
- XOR operators, when the related causes are mutually exclusive.

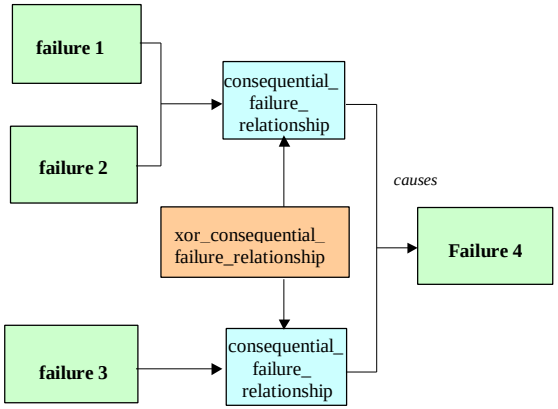


Figure 9 – consequential failures

The association between failure and effects may be used to capture additional information such as probability and safety hazard severity (e.g. critical, minor, none).

- *Detection Method*: the detection method describes how the operator or the maintenance technician detects a specific failure. Warning devices, test equipment, and their normal or abnormal indication are described by the detection methods.

6.5.3.2 Task Analysis

The basic objective of the task analysis is to define necessary tasks for the support and

A task is performed in a *support_scenario* that describes under which conditions (environment, national regulations, available skills, etc) the task is expected to be performed.

The diagram in figure 10 describes the task context.

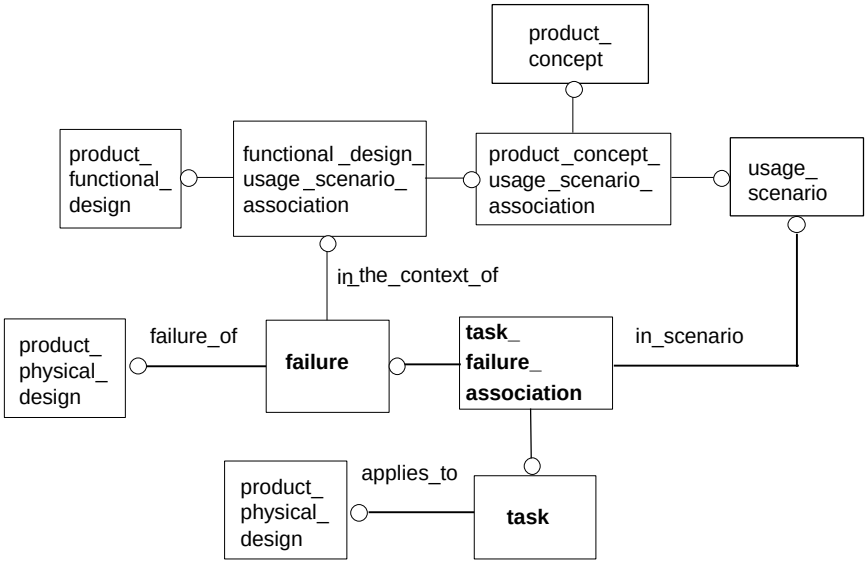


Figure 10 - task basic relationships

A task provides the instructions on how to perform a particular activity or action.

Some information requirements related to *task* are the followings:

1. *task*(s) may be classified. Servicing Tasks, Scheduled Tasks, Occasional Tasks are examples of task classes;
2. a *task* may relate to another *task* for a particular reason;
3. the possible rationale for two *task*(s) to be related should be defined. Alternate *task*(s) is an example of this rationale;

7. simple conditions may be combined with AND, OR and XOR logical operators to define more complex conditions (e.g. do task 'A' IF date(current)-date(task_A_last_done)>90 .OR. mileage(current) – mileage(task_A_last_done) > 3000);

8. condition monitoring may require that certain parameters of *product_instance* and *support_scenario* be measured and recorded. Where data is collected automatically the source sensor[23] should be identified.

A task is usually decomposed in elementary task stages or steps that may be seen as modular building blocks. Tasks may be defined by assembling the elementary (base) stages using selected criteria.

Some requirements for *base_stage* are the followings:

1. a *base_stage* may be assigned to zero, one or many *task*(s). This implies that a *base_stage* may exist without a *task*[24] and that the same *base_stage* may be assigned to many *task*(s);

2. a *base_stage* may be either a *method_stage* (what to do) or an *advisory_stage* (warnings, cautions, …);

3. a *task* shall include at least one *method_stage*;

4. a *method_stage* may be described in different forms, e.g. by a simple narrative description of what to do or by more sophisticated forms of representations such as video, audio, virtual reality;

5. some resources[25] may be needed to perform the activity described by a *method_stage*. Resources may have a role[26] (e.g. spare parts, consumables, test equipment, calibration equipment, etc.) and may be quantified;

6. the identified resources include the following:

   - *facility_or_infrastructure*: this may be a reference to a generic facility (e.g. 220V power supply) or a generic infrastructure (e.g. a dry-dock). It also may be a reference to a specific named and located facility;
   - *information_requirement*: a reference to the applicable information

- *build in facilities* in the existing product;
- *consumables* (e.g. oil, water ..)

Having defined the entities *task* and *base_stage*, there is still the need to define how the *base_stage*(s) are assembled together to form a *task*.

Basically, a task may be seen as a linear flow or sequence of *base_stage*(s). In such simple instance, Task 'A' is made up of Step '1' (the *base_stage*) followed by Step '2' and so on.

More frequently, however, the flow of actions is not a plain linear sequence of *base_stage*(s). For example, the flow of actions (what to do next) may depend on the results of a test condition.

A flow control structure that is external both to *task* and to *base_stage* may be used to alter the flow of actions of a task. Use of a control structure would enable Interactive Electronic Technical Manual (ITEM) style functionality to be provided directly from the Product Life Cycle Support database. The constructs that should be supported by the control structure are the followings:

1. *base_stage_sequence*: it is the list of *base_stage*(s) to be carried out in order;
2. *control_transfer*: rather than referencing a *base_stage*, the control structure is allowed to call another *base_stage_sequence* or even another *task*. This means that tasks and sequences can be nested within each other;
3. *conditional_transfer*: enables a choice between different routes depending on the result of a test condition. The choice could be between two alternatives (IF … ELSE … ENDIF) or between many (DO CASE .. CASE … ENDCASE);
4. *looping_method*: enables one or more *base_stage*(s), *base_stage_sequence*(s) or *task*(s) to be repeated. There are three ways of controlling the numbers of repeats and these can be combined:
   - *repeat_count*: repeat a specified number of times (FOR … NEXT);
   - *repeat_until*: repeat until a test condition is met (DO UNTIL …

6.6 Product Instance

A product_instance is the physical realization, through the manufacturing process, of a product_physical_design. In this paragraph the term *product_instance* refers to a specific physical object (e.g. Helicopter with tail number 97-01).

Some detailed requirements for product_instance are the following:

1. a *product_instance* is the physical realization of zero or one *product_design*. The relationship between product_instance and design is optional since it will not always be possible or necessary to establish this relationship;
2. a *product_instance* is owned[27] by exactly one[28] organization at a given point in time. Ownership may change over time during the life-cycle. The capability to associate a date and time with an ownership change and to maintain history of ownership shall be provided;
3. a *product_instance* is manufactured by one or more organizations;
4. a *product_instance* is supplied (sold?) by one or more organizations
5. the combination *product_instance* identification and the assigning organization identification will uniquely define the product_instance;
6. a *product_instance* may have a serial number that enables to distinguish it from other product_instance(s) based on the same design;
7. a *product_instance* may have a lot or batch number that enables identification of a group of units of the same design which are manufactured or assembled by one producer under uniform conditions and which are expected to function in the same manner. A block identifier may be assigned to designate a quantity of consecutive production of *product_instance*(s) which will have similar characteristics;
8. a *product_instance* may have both a serial number and a lot number. At least one of the two is necessary to identify the product_instance. If both are assigned, a correlation of serial numbers and lot numbers is to be maintained;

11. possible classes should be defined. Class of *product_instance* may include role, function and condition state (e.g. “in repair”[29], “spare parts”).

12. a *product_instance* structure may be the assembly of other product_instance(s). As a minimum, a product instance structure should include the component *product_instance* identification.

13. a *product_instance* structure may be subject to changes due to a configuration variant or replacement of defective items. The old components (predecessors) and the new components (successors) shall be identified. The capability to associate a date, time and organization responsible for the change with each change implementation and to maintain history and rationale of changes shall be provided;

14. *product_instance* structure changes due to configuration variant shall be referenced by Configuration Change entities such as change_request, implementation directive and applied_solution;

15. a *product_instance* may be controlled by zero, one or many organizations at a given point in time;

16. control of a *product_instance* may have different forms (e.g. operational, logistics, storage, etc.). There is exactly one controlling organization associated with each control form at a given point in time;

17. *product_instance* control may change over time during life-cycle. The capability to associate a date and time with each transfer of control and to maintain history of control shall be provided;

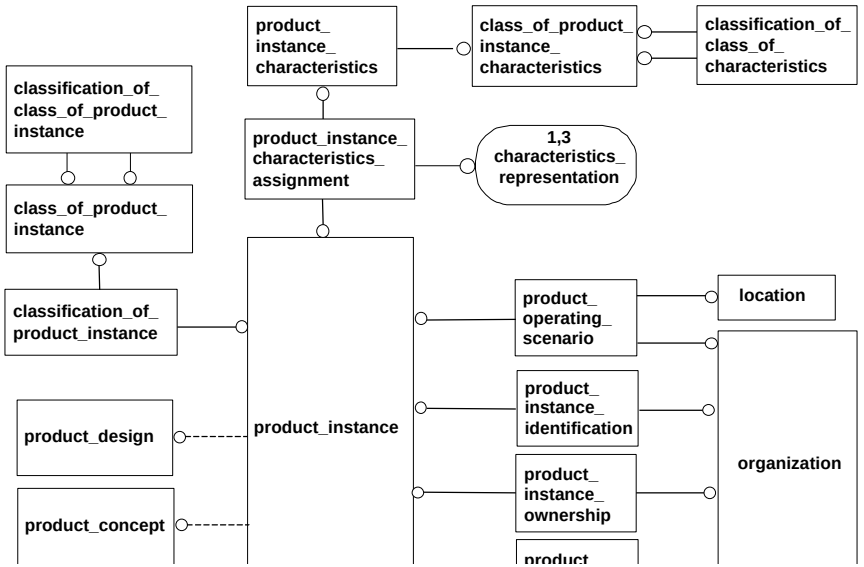
18. a *product_instance* exists at one location at a given point in time[30]. It may be moved from one location to another. The capability to associate a date and time with each change of location and to maintain history of location change shall be provided;

19. a *product_istance* is characterized by the properties defined for the related *product_design*;

- point in time: defined by a date and time;
- period : defined by a time measure.[31]

23. the capability to maintain *product_instance*(s) maintenance history and operational history shall be provided:
    - maintenance history may include date, type, responsible person/organization ;
    - operational history[32] may include running hours, flight hours, shell fired;
24. a *product_instance* may be used as the alternate for another;
25. a *product_instance* operates in one scenario at a given point in time. This relates to the information_objects and tasks defined for that scenario (e.g. maintenance tasks in desert operations);
26. zero, one or many actual functionality may be related to a *product_instance*.

The diagram in figure 11 is a high level model that covers most of the above requirements.



Product_instance(s) have a number of relationships with organizations, these include identification, ownership and control. Ownership relationship is the easiest to understand. In this model an ownership change (e.g. the action of selling) is covered by recording a new ownership relationship and setting the end effectivity for the previous ownership. The current ownership is identified by the relationship which has a date for a start effectivity and has a blank field (empty date) for the end effectivity.

This concept of start effectivity and end effectivity is not shown in the high level model in figure 11 but most of the above relationship make use of it.

A *product_instance* may relate to another *product_instance.* A typical relationship is the *product_instance_assembly* which defines a parent-child relationship between two *product_instance*(s) in an assembly structure. This relationship is used to describe, for example, that the Accelerometer with Serial Number 100267 is part of the Guidance Set with Serial Number 0982701 which is mounted on the SS-01 Missile with the Serial Number 003982-45.

The *product_instance_succession* relationship identifies that one product_instance (predecessor) has been replaced by another product_instance (successor) for an identified reason. This entity may communicate information such as that the Accelerometer with Serial Number 100267 replaced, on 09 July 1998, the Accelerometer with Serial Number 100208 because of an out of tolerance reading during the preventive electronic test.

A *product_instance* may be classified. This classification should not replicate information already defined in the *product_design* classification or *product_design* properties. The fact, for example, that the ETS with Serial Number 018992 is an M09 Electronic Test Equipment used to check the SS-01 Missile Guidance Set is information already available through the relationship to the *product_design* schema. A class of *product_instance*(s) could be, for example, the class ‘in Repair’. This would give the capability to query the database and derive the list of *product_instance*(s) that are under repair. In any case, the possible *product_instance* classes should be constrained in an Agreement of Common Understanding between the partners sharing this information.

A *product_instance* has characteristics. It is important to distinguish between

June1998. It is therefore possible to schedule the next calibration of ETS M09 S.N. 018992 for the end of December 1998.

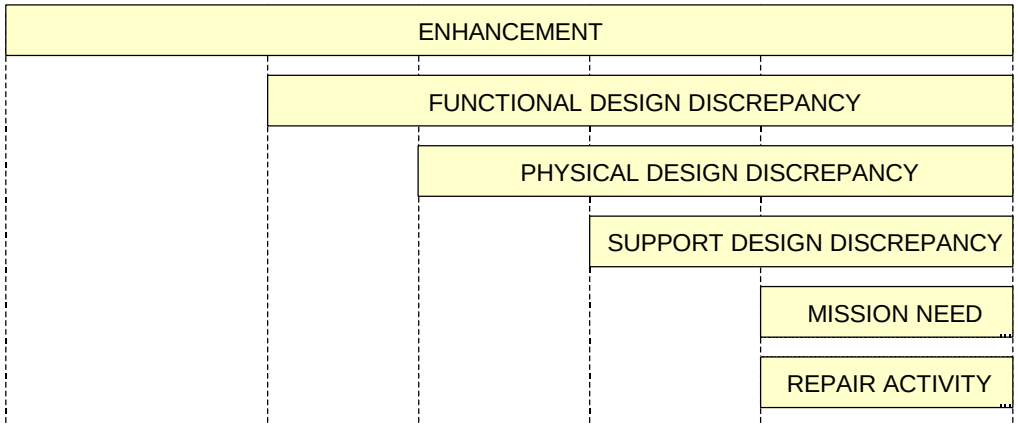
6.7 Product Configuration Change

6.7.1 Change Control Basics

A need for change to the baseline configuration may arise from[33]:

- an enhancement: a new or improved *product_requirement*;
- a discrepancy: *product_instance*(s) based on the same design that fails to meet one or more *product_requirement*. A discrepancy may be a functional design, physical design or a support design discrepancy;
- a mission need: a *product_instance* that is reconfigured, in one of the permissible configurations, for a mission;
- a repair activity: a *product_instance* defective component[34] is replaced by a spare part (the *product_instance* structure is changed).

The configuration change activity results in the creation of new data instances, such as new part/assembly identifications and new relationships. Diagram in figure 12 illustrates, for each of the above triggers for change, the product objects that are affected.



(2) a support discrepancy may result in new support design data and new *product_instance/product_design* relationships;

(3) a configuration change for a mission need may create a new *product_instance* structure but would not affect *product_requirement* and *product_design*.

6.7.1.1 Change Process

The key components of configuration change are: (1) request, (2) implementation directive[35], (3) actual resolution.

The following are some data requirements to support the change process:

1. the change process normally starts with a *request* that could be either a request for initial issue or a change request;
2. a change *request* may be either a request for enhancement or a request to correct or accept a discrepancy[36];
3. a *request* for enhancement may be triggered by a new, user defined, *usage_scenario* or a new user *product_requirement*.
4. a request is submitted by an organization at a certain point in time (date and time);
5. the requesting organization assigns an identifier to the request. The identifier is to be unique within the organization domain. The combination of the requesting organization identifier and the request identifier will uniquely define the request;
6. the request identifies either the baseline *product_concept*, *product_requirement*, *product_design* and/or *product_instance* that are impacted by the request;
7. an initial issue request should address a product_concept and its product_requirement baseline;
8. an enhancement related change request should address a *product_requirement* or *product_design* and may address a *product_instance* baseline;
9. a discrepancy related change request should address either a *product_design* or a

time of approval; (d) approval organization and organization role; (e) relationship with other approvals;

12. an approved request may be referenced by zero, one or many *implementation_directive*[38];

13. an *implementation_directive* describes the resolution applied to the related request. It may include description of tasks, manpower, facilities, parts, kits, support equipment, money needed to apply the actual resolution. It also identifies the organization responsible for applying the actual resolution and the timetable for finalization;

14. an *implementation_directive* may address one or many approved requests;

15. an *implementation_directive* is prepared by an organization at a certain point in time;

16. the combination of the *implementation_directive* identifier and the organization identifier of the organization assigning the *implementation_directive* identifier will uniquely define the *implementation_directive*;

17. an *implementation_directive* may impact different configuration baseline items from those identified in the request[39] (see point 6 above);

18. *implementation_directive*(s) may be classified, decomposed and aggregated;

19. an *implementation_directive* may be referenced by zero, one or many approvals[40]. The approval related information are those listed at point 11 above;

20. an approved implementation_directive may be referenced by zero, one or many *actual_resolution*;

21. an *actual_resolution* may address one approved implementation directive;

22. from the information management perspective, an *actual_resolution* results in the creation of new data instances (e.g. new design version, new product_instance structure, new relationships between product views, etc);

23. an *actual_resolution* is applied by a responsible organization;

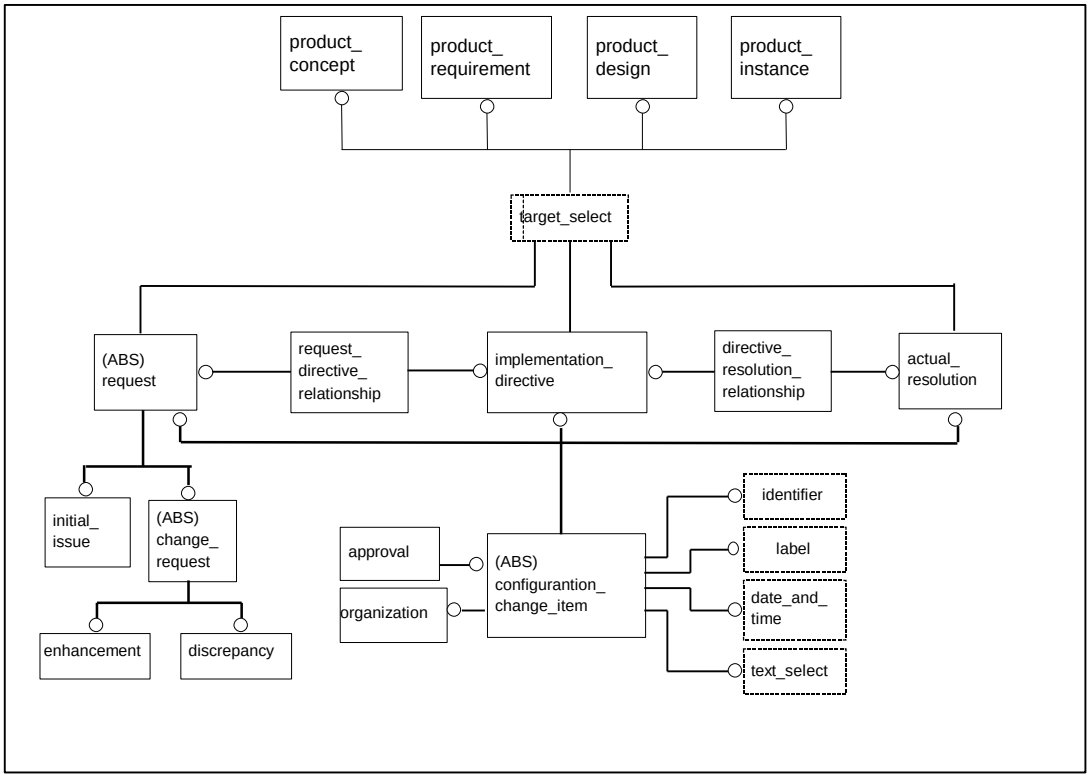


Figure 13 – High level diagram for Change Control

6.8 Other Issues to Consider

6.8.1 Supportability Characteristics

- Required
- Actual
- Predicted (by whom)
- Calculated (by whom)

- *Pipe-line time* (product_physical_design property);
- *Shelf life* (product_physical_design property).
- *Kits/Groups*
- *Supply quantities*

6.8.3 Packaging

- Package (a class of product_physical_design instance);
- Package physical characteristics (product_physical_design property):
  - Size;
  - Weight;
  - Geometry/dimensions;
  - Stacking requirements;
  - Protection;
  - Marking;
  - Safety/hazard;
- Package Identifier and Label(product_physical_design identification);
- Package Content (product_physical_design collection);

6.8.4 Handling

- Handling equipment (a class of product_physical_design instance);
- Handling instructions (task description);
- Safety and security during handling(task advisory_stage description).
- Emergency actions.

6.8.5 Storage

- Type of storage
- Permissible Storage condition (temperature, humidity)
- Storage methods (stacking and distances);
- In-storage maintenance;
- Safety and security during storage;

Annex 1: Relevant existing material

This list seeks to identify what existing documentation should be considered in developing the requirement. The material will be collected by the NCO either on paper, or in the NCO Office IT system (g/2/6/4/PLCS Background)

PWI Draft Interim report (1998-03-01)
NATO CALS Through Life Business Model, Version 6
SwedCALS CALS Technical Goal 2
- NATO CALS Data Model and Rig Test Report
- NATO CALS Pilot Project #1, Task 2.4 function list

POSC Caesar CD
NCFP Reports:
- Data Architecture and Interchange Specification
- NCFP Interchange Candidates List

Minutes from WG3/T8 (Florence, Orlando and Bad Aibling)
Other Bad Aibling Material:
- Notes on Wednesday CM session
- JD Notes on product id session (more to be sought from Gerry Radick and Bob Fisher)
- Jim Kindrick presentation on product id.
- JK PDM modules presentation J
- JD Notes on above

New version(s) of PDM schema modules (through web)
STEP Shipbuilding Common Modules?
Mil Spec CM2549
AP208
AP203
New IRs part 41,43,44
NATO CALS Acquisition Workshop Report – data model
ISO/TC184 Supplementary QC Directives (from pdt web site)
IMS project proposal form.
Functional Guide for Support Information System (France MoD – in French)
US Mil-PRF 49506
Rome airforce base home page!

Annex 2: Glossary of Terms

Annex 3; Industry Classification

The industrial classification codes used for bounding the work are taken from KOMPASS [1].

Note: The mining/quarrying; basic metal and component manufacturing; construction; and electricity, gas, water and chemical production industries are not included.

Code	Description
35 - 36	Metal industry
37	Electrical, electronic, data processing and nucleonic equipment
38	Precision equipment; measuring, testing, optical, photographic, cinematographic, medical and surgical equipment
39	Transport equipment, infrastructure
40	Hydraulic and pneumatic equipment, steam machines, engines, pumps, compressors, refrigeration, heating and air conditioning equipment
41	Agriculture, horticulture and forestry equipment, food, drink and tobacco equipment
42	Chemical, rubber and plastics plant and equipment. Mechanical preparation of materials. Collecting and processing equipment for industrial and domestic refuge. Water treatment equipment. Packing machinery
43	Textile, clothing, leather industry and shoemaking equipment
44	Pulp and paper industry. Printing office machinery and equipment
45	Mining and quarrying, oil and gas extraction equipment. Stone and earth, ceramic and glass industry equipment. Mechanical handling equipment. Road making, building, offshore and underwater equipment
46	Heavy industry and metal working plant and machinery
47	Metal and woodworking machines, machine tools and accessories, special purpose machines. Industrial robots
48	General mechanical engineering sub-contractors

Table 1: Table of industrial classifications

[2] REED Information Services (In association with the CBI), Windsor Court, East Grinstead House, East Grinstead, West Sussex, RH19 1XA. **KOMPASS, Volume I - Products and Services (UK),** 1994

List of issues

1. Is effectivity still an issue?

   Probably not. Most of the effectivity relationships will be managed within the model as relationships between entities. Where needed additional effectivity relationships, not modeled directly, can be established by using the STEP "effectivity" link.

2. To what degree should inventory holdings be part of PLCS.

   The scope of "Manage Inventory" needs more work. It is clear that PLCS should model the product properties required to operate to the logistic supply chain (e.g. product weight, transportability characteristic, required packaging etc.) It is also likely that the "static" properties of the supply chain itself would need modelling to support LSA (e.g. planned storage locations, max/min stock holdings). Opinions differed over the practicality of modelling the behaviour of the supply system (e.g. asset tracking, how long will this part take to arrive?), although this clearly will have an impact on the maintenance activity. General view was that supply system behaviour would fall within scope, but for attention later in the project.

3. Are there other standards dealing with inventory management?

4. Does PLCS extend beyond a "single system"? What is meant by "FLEET Issues"

   The "Fleet" issue are now addressed in the STR – issue resolved?.

5. Are there extra information requirements arising from the need to AUDIT support history and maintain product traceability